SSCM Exercise 6

Nikolaus Czernin - 11721138

```
library("tidyverse")
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(knitr)
library(glmnet)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-7
```

```
library("ISLR")
# Custom print function
print_ <- function(...) print(paste(...))

set.seed(11721138)
```

```
soft <- function(a, b) {
  sign(a) * max(abs(a) - b, 0)
}
```

This function performs soft thresholding on the two inputs, according to this formula:
$soft(a, b) = max(0, (sign(a) * (|a| - b)))$.

Task 1

Defining a function for the shooting alrogithm.

```
lasso_shooting <- function(X, y, lambda, tolerance_limit=1e-07, max_iter=1e+05, verbose=F){
  n <- nrow(X)
```

```r
  p <- ncol(X)
  # do normal least squares estimation to get starting coefficients
  # remove intercept for scaled data
  baseline_coeffs <- coef(lm(y ~ ., data = as.data.frame(X)))
  # exluce the intercept from beta
  beta <- baseline_coeffs %>% .[-1]
  beta0 <- baseline_coeffs %>% .[1]

  # center the y_data by the intercept
  y.centered <- y - beta0

  if (verbose)print("Starting coefficients:")
  if (verbose)print(beta)
  # iterate over the maximum iterations,
  # this is like a while loop with automatic max iter stopping
  for (iter in 1:max_iter) {
    # rename the previouscoefficients
    beta_previous <- beta
    # now we iterate over all coefficients and update them
    for (j in 1:p){
      # get aj
      aj <- 2 * sum(X[,j] ^ 2)
      # get cj
      cj <- 2 * sum(X[,j] * (y.centered - X %*% beta_previous + beta_previous[j] * X[,j]))
      # apply soft thresholding to get a new beta
      # this is the jth coefficient in the new beta
      beta[j] <- soft(cj/aj, lambda/aj)
    }
    # check i fstopping cirterion is met
    if (sum(abs(beta - beta_previous)) < tolerance_limit) {
      if (verbose)print_("Stopping early at iteration", iter)
      break
    }
  }
  # put the intercept back into the coefficients
  beta <- c(beta0, beta)
  beta
}
```

The function first does normal least squares fitting on the data to get the initial coefficients to regularize, using the lm() function. I then remove the intercept from the initial coefficients and substract it from the response, centering it around the intercept. Then I create a for loop, iterating for the maximum iterations passed as a parameter.
Inside, I iterate over p, i.e. each coefficient $beta_j$. I compute both variables a_j and c_j and apply the soft-threshold-function defined above to get the new coefficient $beta_j$. After each outer iteration, after getting all new betas, I check, if the sum of changes of the coefficients is small enough to stop early.
Finally, I reappend the intercept to the new coefficients.

The default values for the tolerance limit and maximum number of iterations is the same values that glmnet uses as default parameters too.

Defining a function to test different lambda values

```
lambdas <- exp(seq(log(1e-4), log(1e+2), length.out = 100))

get_lasso_coeffs <- function(X, y, lambdas, tolerance_limit=1e-07, max_iter=1e+05, verbose=F){
  lambdas %>%
    sapply( function(l) lasso_shooting(X, y, l, tolerance_limit, max_iter, verbose)) %>%
    t() %>%
    as.data.frame() %>%
    mutate(lambda = lambdas) %>%
    select(lambda, everything())
}
```

In this function I iterate over the given lambdas values and apply the lasso function defined above to the lambda values and the other parameters. I then only transpose the output and add the lambda values to get a nice dataframe finally.

Comparing our implementation with glmnet

Generating sample data

```
n <- 100  # num of observations
p <- 10   # num of variables

# generate variables
# start with random noise in a matrix with the wanted shape
X <- matrix(rnorm(n * p), n, p)
# now generate some random integers, they will be the "correct" coefficients
beta_true <- sample(seq(-3, 3), p, replace = TRUE)
# now do matrix multiplication with the coefficients and the features, then add more random noise
y <- X %*% beta_true + rnorm(n)
# for the lambdas, get 20 values between 0 and 1
lambdas <- seq(-4, 1, 0.05) %>% exp()
```

```
lasso_coeffs <- get_lasso_coeffs(X, y, lambdas)
lasso_coeffs %>% head(5)
```

```
##       lambda (Intercept)       V1       V2       V3          V4         V5
## 1 0.01831564 -0.05457108 1.981016 1.105321 3.047523 -0.02192543 0.01416726
## 2 0.01925470 -0.05457108 1.981009 1.105314 3.047516 -0.02192098 0.01416171
## 3 0.02024191 -0.05457108 1.981003 1.105308 3.047509 -0.02191629 0.01415589
## 4 0.02127974 -0.05457108 1.980996 1.105301 3.047502 -0.02191137 0.01414976
## 5 0.02237077 -0.05457108 1.980988 1.105293 3.047494 -0.02190619 0.01414332
##         V6       V7         V8        V9       V10
## 1 2.050718 1.087100 0.06779779 -3.010714 -2.970455
## 2 2.050716 1.087094 0.06779158 -3.010706 -2.970454
## 3 2.050714 1.087086 0.06778506 -3.010698 -2.970453
## 4 2.050711 1.087079 0.06777821 -3.010689 -2.970451
## 5 2.050709 1.087071 0.06777100 -3.010680 -2.970450
```

```
lasso_coeffs %>% tail(5)
```

```
##       lambda (Intercept)       V1       V2       V3           V4           V5
## 97  2.225541 -0.05457108 1.966131 1.090102 3.031599 -0.011452942 0.0011386694
## 98  2.339647 -0.05457108 1.965361 1.089315 3.030775 -0.010911550 0.0004651368
## 99  2.459603 -0.05457108 1.964561 1.088530 3.029932 -0.010342292 0.0000000000
```

```
## 100 2.585710 -0.05457108 1.963737 1.087790 3.029088 -0.009743628 0.0000000000
## 101 2.718282 -0.05457108 1.962871 1.087012 3.028202 -0.009114270 0.0000000000
##           V6       V7         V8        V9       V10
## 97  2.045509 1.070864 0.05321566 -2.992103 -2.967542
## 98  2.045240 1.070024 0.05246181 -2.991141 -2.967392
## 99  2.044932 1.069178 0.05165552 -2.990147 -2.967182
## 100 2.044560 1.068361 0.05078013 -2.989139 -2.966859
## 101 2.044169 1.067502 0.04985986 -2.988080 -2.966519
```

```
fit_glmnet <- glmnet(X, y, alpha=1, lambda=lambdas)

get_glmnet_coeffs <- function(X, y, lambdas, custom_colnames=NULL){
  out <- fit_glmnet %>%
    coef() %>%
    t() %>%
    as.matrix() %>%
    as.data.frame() %>%
    mutate(lambda = lambdas) %>%
    select(lambda, everything())
  rownames(out) <- NULL
  if (!is.null(custom_colnames)) colnames(out) <- custom_colnames
  out
}
custom_colnames <- colnames(lasso_coeffs)
glmnet_coeffs <- get_glmnet_coeffs(X, y, lambdas, custom_colnames)
glmnet_coeffs %>% head(5)
```

```
##       lambda (Intercept) V1 V2        V3 V4 V5         V6 V7 V8          V9
## 1 0.01831564  -0.7550038  0  0 0.1560607  0  0 0.00000000  0  0  0.00000000
## 2 0.01925470  -0.7522988  0  0 0.2761331  0  0 0.01606127  0  0  0.00000000
## 3 0.02024191  -0.7511982  0  0 0.3806511  0  0 0.14247654  0  0  0.00000000
## 4 0.02127974  -0.7388097  0  0 0.4917592  0  0 0.25007137  0  0 -0.08773478
## 5 0.02237077  -0.7215948  0  0 0.6030581  0  0 0.34635104  0  0 -0.21319468
##          V10
## 1 -0.5726738
## 2 -0.6997152
## 3 -0.8273290
## 4 -0.9506092
## 5 -1.0687806
```

```
glmnet_coeffs %>% tail(5)
```

```
##       lambda (Intercept)       V1       V2       V3           V4 V5       V6
## 97  2.225541 -0.06099217 1.949108 1.076220 3.015162 -0.001970228  0 2.039620
## 98  2.339647 -0.06062127 1.950655 1.077519 3.016685 -0.002938027  0 2.040234
## 99  2.459603 -0.06026835 1.952126 1.078757 3.018136 -0.003858543  0 2.040817
## 100 2.585710 -0.05993260 1.953526 1.079935 3.019516 -0.004734142  0 2.041372
## 101 2.718282 -0.05961322 1.954857 1.081056 3.020828 -0.005567033  0 2.041899
##           V7         V8        V9       V10
## 97  1.053475 0.03673627 -2.973403 -2.962781
## 98  1.055025 0.03829961 -2.975186 -2.963302
## 99  1.056500 0.03978789 -2.976883 -2.963797
## 100 1.057902 0.04120377 -2.978497 -2.964267
## 101 1.059237 0.04255063 -2.980033 -2.964715
```

The previous two outputs print the coefficients of the sample data after first using my own lasso function and then using glmnet(). The former reduces sometimes more variable down to zero than our shooting algorithm function.

Now, lets compare their evaluation performances on the training data.

```
MSE <- function(y, yhat){
  mean((y - yhat)^2)
}
```

This function computes the MSE of a model by taking its true response and predictions.

```
make_prediction <- function(X, beta){
  # to include the intercept, add an identity column to X
  cbind(1, X) %*% beta
}
```

This function takes the feature matrix X and multiplys it with the model coefficients, which produces the response. The coefficients include the intercept, so they have 1 value more than there are columns in the feature matrix, so you need to add a column of 1s to the beginning of the feature matrix.

```
evaluate <- function(y, X, beta){
  MSE(y, make_prediction(X, beta))
}
```

This function just combines the two defined above.

```
lasso_coeffs$MSE <- apply(lasso_coeffs, 1, function(row) evaluate(y, X, row[-1]))
glmnet_coeffs$MSE <- apply(glmnet_coeffs, 1, function(row) evaluate(y, X, row[-1]))
lasso_coeffs
```

```
##          lambda (Intercept)       V1       V2       V3           V4           V5
## 1    0.01831564 -0.05457108 1.981016 1.105321 3.047523 -0.021925433 0.0141672552
## 2    0.01925470 -0.05457108 1.981009 1.105314 3.047516 -0.021920978 0.0141617126
## 3    0.02024191 -0.05457108 1.981003 1.105308 3.047509 -0.021916294 0.0141558858
## 4    0.02127974 -0.05457108 1.980996 1.105301 3.047502 -0.021911370 0.0141497603
## 5    0.02237077 -0.05457108 1.980988 1.105293 3.047494 -0.021906193 0.0141433206
## 6    0.02351775 -0.05457108 1.980981 1.105285 3.047485 -0.021900752 0.0141365509
## 7    0.02472353 -0.05457108 1.980973 1.105277 3.047477 -0.021895031 0.0141294340
## 8    0.02599113 -0.05457108 1.980964 1.105268 3.047468 -0.021889016 0.0141219522
## 9    0.02732372 -0.05457108 1.980955 1.105259 3.047458 -0.021882694 0.0141140869
## 10   0.02872464 -0.05457108 1.980946 1.105249 3.047448 -0.021876047 0.0141058182
## 11   0.03019738 -0.05457108 1.980936 1.105239 3.047437 -0.021869059 0.0140971196
## 12   0.03174564 -0.05457108 1.980925 1.105228 3.047426 -0.021861713 0.0140879811
## 13   0.03337327 -0.05457108 1.980914 1.105217 3.047414 -0.021853990 0.0140783740
## 14   0.03508435 -0.05457108 1.980903 1.105205 3.047402 -0.021845872 0.0140682743
## 15   0.03688317 -0.05457108 1.980891 1.105193 3.047389 -0.021837337 0.0140576568
## 16   0.03877421 -0.05457108 1.980878 1.105180 3.047375 -0.021828365 0.0140464950
## 17   0.04076220 -0.05457108 1.980864 1.105166 3.047361 -0.021818933 0.0140347608
## 18   0.04285213 -0.05457108 1.980850 1.105152 3.047346 -0.021809017 0.0140224251
## 19   0.04504920 -0.05457108 1.980836 1.105137 3.047330 -0.021798592 0.0140094568
## 20   0.04735892 -0.05457108 1.980820 1.105121 3.047313 -0.021787634 0.0139958237
## 21   0.04978707 -0.05457108 1.980804 1.105104 3.047296 -0.021776113 0.0139814916
## 22   0.05233971 -0.05457108 1.980786 1.105086 3.047277 -0.021764002 0.0139664247
## 23   0.05502322 -0.05457108 1.980768 1.105068 3.047258 -0.021751270 0.0139505853
## 24   0.05784432 -0.05457108 1.980749 1.105048 3.047238 -0.021737884 0.0139339281
## 25   0.06081006 -0.05457108 1.980729 1.105028 3.047216 -0.021723812 0.0139164225
## 26   0.06392786 -0.05457108 1.980708 1.105006 3.047194 -0.021709020 0.0138980194
## 27   0.06720551 -0.05457108 1.980686 1.104984 3.047170 -0.021693468 0.0138786728
```

```
## 28  0.07065121 -0.05457108 1.980663 1.104960 3.047145 -0.021677120 0.0138583342
## 29  0.07427358 -0.05457108 1.980638 1.104935 3.047119 -0.021659933 0.0138369529
## 30  0.07808167 -0.05457108 1.980613 1.104909 3.047092 -0.021641865 0.0138144753
## 31  0.08208500 -0.05457108 1.980586 1.104881 3.047063 -0.021622871 0.0137908452
## 32  0.08629359 -0.05457108 1.980557 1.104852 3.047032 -0.021602903 0.0137660037
## 33  0.09071795 -0.05457108 1.980528 1.104822 3.047001 -0.021581911 0.0137398884
## 34  0.09536916 -0.05457108 1.980496 1.104790 3.046967 -0.021559842 0.0137124342
## 35  0.10025884 -0.05457108 1.980463 1.104756 3.046932 -0.021536643 0.0136835724
## 36  0.10539922 -0.05457108 1.980429 1.104721 3.046895 -0.021512254 0.0136532309
## 37  0.11080316 -0.05457108 1.980392 1.104683 3.046856 -0.021486613 0.0136213278
## 38  0.11648416 -0.05457108 1.980354 1.104644 3.046815 -0.021459659 0.0135877949
## 39  0.12245643 -0.05457108 1.980314 1.104603 3.046772 -0.021431323 0.0135525427
## 40  0.12873490 -0.05457108 1.980271 1.104560 3.046726 -0.021401534 0.0135154831
## 41  0.13533528 -0.05457108 1.980227 1.104514 3.046679 -0.021370217 0.0134765234
## 42  0.14227407 -0.05457108 1.980180 1.104466 3.046629 -0.021337295 0.0134355662
## 43  0.14956862 -0.05457108 1.980131 1.104416 3.046576 -0.021302685 0.0133925091
## 44  0.15723717 -0.05457108 1.980079 1.104363 3.046521 -0.021266301 0.0133472444
## 45  0.16529889 -0.05457108 1.980025 1.104308 3.046462 -0.021228051 0.0132996589
## 46  0.17377394 -0.05457108 1.979967 1.104249 3.046401 -0.021187840 0.0132496336
## 47  0.18268352 -0.05457108 1.979907 1.104188 3.046337 -0.021145567 0.0131970435
## 48  0.19204991 -0.05457108 1.979844 1.104123 3.046269 -0.021101127 0.0131417571
## 49  0.20189652 -0.05457108 1.979778 1.104055 3.046198 -0.021054408 0.0130836361
## 50  0.21224797 -0.05457108 1.979708 1.103984 3.046124 -0.021005295 0.0130225351
## 51  0.22313016 -0.05457108 1.979635 1.103909 3.046045 -0.020953662 0.0129582956
## 52  0.23457029 -0.05457108 1.979557 1.103830 3.045963 -0.020899383 0.0128907683
## 53  0.24659696 -0.05457108 1.979476 1.103747 3.045876 -0.020842320 0.0128197788
## 54  0.25924026 -0.05457108 1.979391 1.103660 3.045785 -0.020782332 0.0127451496
## 55  0.27253179 -0.05457108 1.979301 1.103568 3.045689 -0.020719269 0.0126666940
## 56  0.28650480 -0.05457108 1.979207 1.103472 3.045588 -0.020652972 0.0125842160
## 57  0.30119421 -0.05457108 1.979108 1.103371 3.045482 -0.020583276 0.0124975092
## 58  0.31663677 -0.05457108 1.979004 1.103264 3.045371 -0.020510007 0.0124063569
## 59  0.33287108 -0.05457108 1.978895 1.103152 3.045254 -0.020432981 0.0123105310
## 60  0.34993775 -0.05457108 1.978779 1.103034 3.045130 -0.020352006 0.0122097921
## 61  0.36787944 -0.05457108 1.978658 1.102911 3.045001 -0.020266879 0.0121038882
## 62  0.38674102 -0.05457108 1.978531 1.102781 3.044865 -0.020177388 0.0119925545
## 63  0.40656966 -0.05457108 1.978398 1.102644 3.044722 -0.020083308 0.0118755126
## 64  0.42741493 -0.05457108 1.978257 1.102500 3.044571 -0.019984404 0.0117524639
## 65  0.44932896 -0.05457108 1.978109 1.102349 3.044413 -0.019880430 0.0116231123
## 66  0.47236655 -0.05457108 1.977954 1.102190 3.044247 -0.019771125 0.0114871287
## 67  0.49658530 -0.05457108 1.977790 1.102023 3.044072 -0.019656215 0.0113441730
## 68  0.52204578 -0.05457108 1.977619 1.101848 3.043889 -0.019535415 0.0111938879
## 69  0.54881164 -0.05457108 1.977438 1.101663 3.043696 -0.019408420 0.0110358974
## 70  0.57694981 -0.05457108 1.977249 1.101469 3.043493 -0.019274915 0.0108698066
## 71  0.60653066 -0.05457108 1.977049 1.101265 3.043279 -0.019134564 0.0106952002
## 72  0.63762815 -0.05457108 1.976839 1.101051 3.043055 -0.018987018 0.0105116415
## 73  0.67032005 -0.05457108 1.976619 1.100825 3.042819 -0.018831907 0.0103186716
## 74  0.70468809 -0.05457108 1.976387 1.100588 3.042571 -0.018668843 0.0101158078
## 75  0.74081822 -0.05457108 1.976143 1.100339 3.042310 -0.018497418 0.0099025431
## 76  0.77880078 -0.05457108 1.975887 1.100077 3.042036 -0.018317205 0.0096783440
## 77  0.81873075 -0.05457108 1.975618 1.099802 3.041748 -0.018127751 0.0094426443
## 78  0.86070798 -0.05457108 1.975335 1.099513 3.041445 -0.017928584 0.0091948657
## 79  0.90483742 -0.05457108 1.975037 1.099208 3.041127 -0.017719206 0.0089343832
## 80  0.95122942 -0.05457108 1.974724 1.098888 3.040792 -0.017499092 0.0086605455
## 81  1.00000000 -0.05457108 1.974396 1.098552 3.040440 -0.017267693 0.0083726678
```

```
## 82  1.05127110 -0.05457108 1.974050 1.098199 3.040071 -0.017024430 0.0080700303
## 83  1.10517092 -0.05457108 1.973686 1.097827 3.039682 -0.016768695 0.0077518763
## 84  1.16183424 -0.05457108 1.973304 1.097436 3.039273 -0.016499848 0.0074174101
## 85  1.22140276 -0.05457108 1.972903 1.097026 3.038843 -0.016217217 0.0070657956
## 86  1.28402542 -0.05457108 1.972480 1.096594 3.038391 -0.015920095 0.0066961533
## 87  1.34985881 -0.05457108 1.972036 1.096140 3.037916 -0.015607739 0.0063075591
## 88  1.41906755 -0.05457108 1.971570 1.095663 3.037417 -0.015279369 0.0058990412
## 89  1.49182470 -0.05457108 1.971079 1.095161 3.036892 -0.014934162 0.0054695782
## 90  1.56831219 -0.05457108 1.970563 1.094634 3.036340 -0.014571257 0.0050180961
## 91  1.64872127 -0.05457108 1.970021 1.094079 3.035760 -0.014189744 0.0045434601
## 92  1.73325302 -0.05457108 1.969451 1.093496 3.035150 -0.013788671 0.0040444949
## 93  1.82211880 -0.05457108 1.968851 1.092884 3.034509 -0.013367035 0.0035199473
## 94  1.91554083 -0.05457108 1.968221 1.092239 3.033835 -0.012923781 0.0029685055
## 95  2.01375271 -0.05457108 1.967559 1.091562 3.033127 -0.012457801 0.0023887907
## 96  2.11700002 -0.05457108 1.966863 1.090850 3.032382 -0.011967929 0.0017793533
## 97  2.22554093 -0.05457108 1.966131 1.090102 3.031599 -0.011452942 0.0011386694
## 98  2.33964685 -0.05457108 1.965361 1.089315 3.030775 -0.010911550 0.0004651368
## 99  2.45960311 -0.05457108 1.964561 1.088530 3.029932 -0.010342292 0.0000000000
## 100 2.58570966 -0.05457108 1.963737 1.087790 3.029088 -0.009743628 0.0000000000
## 101 2.71828183 -0.05457108 1.962871 1.087012 3.028202 -0.009114270 0.0000000000
##           V6       V7         V8        V9       V10       MSE
## 1   2.050718 1.087100 0.06779779 -3.010714 -2.970455 0.8964732
## 2   2.050716 1.087094 0.06779158 -3.010706 -2.970454 0.8964732
## 3   2.050714 1.087086 0.06778506 -3.010698 -2.970453 0.8964732
## 4   2.050711 1.087079 0.06777821 -3.010689 -2.970451 0.8964733
## 5   2.050709 1.087071 0.06777100 -3.010680 -2.970450 0.8964733
## 6   2.050706 1.087062 0.06776342 -3.010670 -2.970448 0.8964733
## 7   2.050703 1.087053 0.06775545 -3.010660 -2.970447 0.8964733
## 8   2.050700 1.087044 0.06774708 -3.010649 -2.970445 0.8964733
## 9   2.050697 1.087034 0.06773828 -3.010638 -2.970443 0.8964733
## 10  2.050694 1.087024 0.06772902 -3.010626 -2.970442 0.8964734
## 11  2.050690 1.087013 0.06771929 -3.010614 -2.970440 0.8964734
## 12  2.050687 1.087002 0.06770906 -3.010601 -2.970438 0.8964734
## 13  2.050683 1.086990 0.06769831 -3.010587 -2.970435 0.8964735
## 14  2.050679 1.086977 0.06768700 -3.010572 -2.970433 0.8964735
## 15  2.050674 1.086964 0.06767512 -3.010557 -2.970431 0.8964735
## 16  2.050670 1.086950 0.06766263 -3.010541 -2.970428 0.8964736
## 17  2.050665 1.086935 0.06764949 -3.010525 -2.970426 0.8964736
## 18  2.050660 1.086920 0.06763569 -3.010507 -2.970423 0.8964737
## 19  2.050655 1.086904 0.06762117 -3.010488 -2.970420 0.8964737
## 20  2.050650 1.086887 0.06760591 -3.010469 -2.970417 0.8964738
## 21  2.050644 1.086869 0.06758987 -3.010448 -2.970414 0.8964738
## 22  2.050638 1.086850 0.06757301 -3.010427 -2.970410 0.8964739
## 23  2.050632 1.086830 0.06755528 -3.010404 -2.970407 0.8964740
## 24  2.050625 1.086810 0.06753664 -3.010381 -2.970403 0.8964741
## 25  2.050618 1.086788 0.06751705 -3.010356 -2.970399 0.8964742
## 26  2.050611 1.086765 0.06749645 -3.010329 -2.970395 0.8964743
## 27  2.050603 1.086741 0.06747479 -3.010302 -2.970391 0.8964744
## 28  2.050595 1.086715 0.06745203 -3.010273 -2.970386 0.8964746
## 29  2.050586 1.086689 0.06742810 -3.010242 -2.970381 0.8964747
## 30  2.050577 1.086661 0.06740294 -3.010210 -2.970376 0.8964749
## 31  2.050568 1.086631 0.06737649 -3.010176 -2.970371 0.8964751
## 32  2.050558 1.086600 0.06734869 -3.010141 -2.970366 0.8964753
## 33  2.050547 1.086568 0.06731946 -3.010103 -2.970360 0.8964755
```

```
## 34  2.050536 1.086534 0.06728873 -3.010064 -2.970354 0.8964757
## 35  2.050525 1.086498 0.06725643 -3.010023 -2.970347 0.8964760
## 36  2.050513 1.086460 0.06722247 -3.009980 -2.970340 0.8964763
## 37  2.050500 1.086420 0.06718676 -3.009934 -2.970333 0.8964767
## 38  2.050487 1.086378 0.06714923 -3.009886 -2.970326 0.8964770
## 39  2.050472 1.086334 0.06710978 -3.009836 -2.970318 0.8964774
## 40  2.050458 1.086288 0.06706830 -3.009783 -2.970310 0.8964779
## 41  2.050442 1.086240 0.06702469 -3.009727 -2.970301 0.8964784
## 42  2.050426 1.086189 0.06697885 -3.009669 -2.970292 0.8964790
## 43  2.050409 1.086135 0.06693066 -3.009607 -2.970282 0.8964796
## 44  2.050390 1.086079 0.06688000 -3.009542 -2.970272 0.8964802
## 45  2.050371 1.086019 0.06682674 -3.009475 -2.970261 0.8964810
## 46  2.050351 1.085957 0.06677075 -3.009403 -2.970250 0.8964818
## 47  2.050330 1.085891 0.06671188 -3.009328 -2.970238 0.8964827
## 48  2.050308 1.085822 0.06665001 -3.009249 -2.970226 0.8964837
## 49  2.050285 1.085750 0.06658495 -3.009166 -2.970213 0.8964849
## 50  2.050261 1.085674 0.06651657 -3.009079 -2.970199 0.8964861
## 51  2.050235 1.085594 0.06644467 -3.008987 -2.970185 0.8964875
## 52  2.050208 1.085510 0.06636909 -3.008890 -2.970170 0.8964890
## 53  2.050180 1.085421 0.06628964 -3.008789 -2.970154 0.8964906
## 54  2.050150 1.085328 0.06620611 -3.008682 -2.970137 0.8964925
## 55  2.050118 1.085230 0.06611830 -3.008570 -2.970120 0.8964945
## 56  2.050085 1.085128 0.06602598 -3.008453 -2.970101 0.8964968
## 57  2.050051 1.085020 0.06592894 -3.008329 -2.970082 0.8964992
## 58  2.050014 1.084906 0.06582692 -3.008198 -2.970062 0.8965020
## 59  2.049976 1.084787 0.06571966 -3.008062 -2.970040 0.8965050
## 60  2.049936 1.084661 0.06560691 -3.007918 -2.970018 0.8965084
## 61  2.049893 1.084529 0.06548838 -3.007766 -2.969994 0.8965121
## 62  2.049849 1.084390 0.06536377 -3.007607 -2.969969 0.8965162
## 63  2.049802 1.084244 0.06523277 -3.007440 -2.969943 0.8965207
## 64  2.049753 1.084091 0.06509505 -3.007264 -2.969915 0.8965257
## 65  2.049701 1.083930 0.06495028 -3.007080 -2.969886 0.8965313
## 66  2.049647 1.083760 0.06479808 -3.006885 -2.969856 0.8965374
## 67  2.049590 1.083582 0.06463808 -3.006681 -2.969824 0.8965441
## 68  2.049529 1.083395 0.06446987 -3.006466 -2.969791 0.8965516
## 69  2.049466 1.083198 0.06429304 -3.006241 -2.969755 0.8965598
## 70  2.049400 1.082991 0.06410715 -3.006004 -2.969718 0.8965690
## 71  2.049330 1.082773 0.06391172 -3.005754 -2.969679 0.8965790
## 72  2.049257 1.082545 0.06370627 -3.005492 -2.969638 0.8965902
## 73  2.049180 1.082304 0.06349029 -3.005216 -2.969595 0.8966025
## 74  2.049098 1.082051 0.06326324 -3.004926 -2.969549 0.8966161
## 75  2.049013 1.081786 0.06302454 -3.004622 -2.969502 0.8966311
## 76  2.048924 1.081506 0.06277361 -3.004302 -2.969452 0.8966477
## 77  2.048829 1.081212 0.06250981 -3.003965 -2.969399 0.8966661
## 78  2.048730 1.080904 0.06223248 -3.003611 -2.969344 0.8966864
## 79  2.048626 1.080579 0.06194094 -3.003239 -2.969285 0.8967088
## 80  2.048517 1.080238 0.06163445 -3.002848 -2.969224 0.8967336
## 81  2.048401 1.079879 0.06131225 -3.002436 -2.969160 0.8967610
## 82  2.048280 1.079502 0.06097352 -3.002004 -2.969092 0.8967913
## 83  2.048153 1.079105 0.06061743 -3.001550 -2.969021 0.8968247
## 84  2.048020 1.078689 0.06024308 -3.001072 -2.968946 0.8968617
## 85  2.047879 1.078250 0.05984954 -3.000570 -2.968868 0.8969026
## 86  2.047731 1.077790 0.05943582 -3.000042 -2.968785 0.8969478
## 87  2.047576 1.077305 0.05900089 -2.999486 -2.968698 0.8969977
```

```
## 88  2.047412 1.076796 0.05854366 -2.998903 -2.968607 0.8970528
## 89  2.047241 1.076261 0.05806299 -2.998289 -2.968511 0.8971138
## 90  2.047060 1.075698 0.05755767 -2.997644 -2.968410 0.8971812
## 91  2.046870 1.075107 0.05702644 -2.996966 -2.968304 0.8972557
## 92  2.046671 1.074485 0.05646798 -2.996254 -2.968192 0.8973380
## 93  2.046461 1.073831 0.05588088 -2.995504 -2.968075 0.8974289
## 94  2.046241 1.073144 0.05526369 -2.994717 -2.967952 0.8975294
## 95  2.046009 1.072422 0.05461484 -2.993889 -2.967822 0.8976405
## 96  2.045765 1.071662 0.05393274 -2.993018 -2.967686 0.8977633
## 97  2.045509 1.070864 0.05321566 -2.992103 -2.967542 0.8978990
## 98  2.045240 1.070024 0.05246181 -2.991141 -2.967392 0.8980489
## 99  2.044932 1.069178 0.05165552 -2.990147 -2.967182 0.8982078
## 100 2.044560 1.068361 0.05078013 -2.989139 -2.966859 0.8983693
## 101 2.044169 1.067502 0.04985986 -2.988080 -2.966519 0.8985478
```

```
glmnet_coeffs
```

```
##         lambda (Intercept)         V1         V2        V3           V4 V5
## 1   0.01831564 -0.75500380 0.00000000 0.00000000 0.1560607  0.0000000000  0
## 2   0.01925470 -0.75229878 0.00000000 0.00000000 0.2761331  0.0000000000  0
## 3   0.02024191 -0.75119819 0.00000000 0.00000000 0.3806511  0.0000000000  0
## 4   0.02127974 -0.73880971 0.00000000 0.00000000 0.4917592  0.0000000000  0
## 5   0.02237077 -0.72159477 0.00000000 0.00000000 0.6030581  0.0000000000  0
## 6   0.02351775 -0.70521947 0.00000000 0.00000000 0.7089278  0.0000000000  0
## 7   0.02472353 -0.68964281 0.00000000 0.00000000 0.8096343  0.0000000000  0
## 8   0.02599113 -0.67482583 0.00000000 0.00000000 0.9054292  0.0000000000  0
## 9   0.02732372 -0.66073148 0.00000000 0.00000000 0.9965522  0.0000000000  0
## 10  0.02872464 -0.64486553 0.01632376 0.00000000 1.0837696  0.0000000000  0
## 11  0.03019738 -0.62020486 0.09538539 0.00000000 1.1687978  0.0000000000  0
## 12  0.03174564 -0.59674547 0.17059699 0.00000000 1.2496874  0.0000000000  0
## 13  0.03337327 -0.57443022 0.24214047 0.00000000 1.3266320  0.0000000000  0
## 14  0.03508435 -0.55320329 0.31019474 0.00000000 1.3998240  0.0000000000  0
## 15  0.03688317 -0.53301162 0.37492996 0.00000000 1.4694464  0.0000000000  0
## 16  0.03877421 -0.51380470 0.43650800 0.00000000 1.5356732  0.0000000000  0
## 17  0.04076220 -0.49553451 0.49508285 0.00000000 1.5986701  0.0000000000  0
## 18  0.04285213 -0.47815538 0.55080097 0.00000000 1.6585946  0.0000000000  0
## 19  0.04504920 -0.46162383 0.60380169 0.00000000 1.7155966  0.0000000000  0
## 20  0.04735892 -0.44589853 0.65421753 0.00000000 1.7698186  0.0000000000  0
## 21  0.04978707 -0.43094017 0.70217456 0.00000000 1.8213961  0.0000000000  0
## 22  0.05233971 -0.41671134 0.74779270 0.00000000 1.8704581  0.0000000000  0
## 23  0.05502322 -0.40473872 0.79288989 0.03693916 1.9253236  0.0000000000  0
## 24  0.05784432 -0.38988493 0.84597244 0.09055558 1.9806808  0.0000000000  0
## 25  0.06081006 -0.37354322 0.90106671 0.13976124 2.0322919  0.0000000000  0
## 26  0.06392786 -0.35799800 0.95349364 0.18670273 2.0814594  0.0000000000  0
## 27  0.06720551 -0.34321092 1.00336370 0.23135505 2.1282291  0.0000000000  0
## 28  0.07065121 -0.32914502 1.05080158 0.27382965 2.1727178  0.0000000000  0
## 29  0.07427358 -0.31576512 1.09592588 0.31423274 2.2150368  0.0000000000  0
## 30  0.07808167 -0.30303777 1.13884944 0.35266535 2.2552918  0.0000000000  0
## 31  0.08208500 -0.29093114 1.17967960 0.38922358 2.2935836  0.0000000000  0
## 32  0.08629359 -0.27941495 1.21851844 0.42399884 2.3300078  0.0000000000  0
## 33  0.09071795 -0.26846042 1.25546310 0.45707809 2.3646557  0.0000000000  0
## 34  0.09536916 -0.25804014 1.29060594 0.48854405 2.3976137  0.0000000000  0
## 35  0.10025884 -0.24812807 1.32403484 0.51847539 2.4289644  0.0000000000  0
## 36  0.10539922 -0.23869941 1.35583340 0.54694697 2.4587860  0.0000000000  0
## 37  0.11080316 -0.22973060 1.38608113 0.57402997 2.4871533  0.0000000000  0
```

```
## 38  0.11648416 -0.22120056 1.41482570 0.59964803 2.5140623  0.0000000000  0
## 39  0.12245643 -0.21308518 1.44219617 0.62415983 2.5397332  0.0000000000  0
## 40  0.12873490 -0.20536558 1.46823193 0.64747711 2.5641526  0.0000000000  0
## 41  0.13533528 -0.19802248 1.49299791 0.66965720 2.5873810  0.0000000000  0
## 42  0.14227407 -0.19103750 1.51655604 0.69075555 2.6094766  0.0000000000  0
## 43  0.14956862 -0.18439319 1.53896523 0.71082492 2.6304946  0.0000000000  0
## 44  0.15723717 -0.17807292 1.56028151 0.72991550 2.6504875  0.0000000000  0
## 45  0.16529889 -0.17206089 1.58055818 0.74807502 2.6695054  0.0000000000  0
## 46  0.17377394 -0.16634208 1.59984595 0.76534889 2.6875957  0.0000000000  0
## 47  0.18268352 -0.16090217 1.61819304 0.78178031 2.7048038  0.0000000000  0
## 48  0.19204991 -0.15572757 1.63564533 0.79741035 2.7211726  0.0000000000  0
## 49  0.20189652 -0.15080534 1.65224647 0.81227811 2.7367431  0.0000000000  0
## 50  0.21224797 -0.14612317 1.66803796 0.82642076 2.7515542  0.0000000000  0
## 51  0.22313016 -0.14166935 1.68305928 0.83987366 2.7656430  0.0000000000  0
## 52  0.23457029 -0.13743275 1.69734801 0.85267046 2.7790446  0.0000000000  0
## 53  0.24659696 -0.13340277 1.71093987 0.86484315 2.7917927  0.0000000000  0
## 54  0.25924026 -0.12956933 1.72386885 0.87642217 2.8039190  0.0000000000  0
## 55  0.27253179 -0.12592285 1.73616727 0.88743647 2.8154539  0.0000000000  0
## 56  0.28650480 -0.12245422 1.74786590 0.89791361 2.8264263  0.0000000000  0
## 57  0.30119421 -0.11915475 1.75899397 0.90787976 2.8368635  0.0000000000  0
## 58  0.31663677 -0.11601620 1.76957932 0.91735986 2.8467917  0.0000000000  0
## 59  0.33287108 -0.11303071 1.77964842 0.92637761 2.8562357  0.0000000000  0
## 60  0.34993775 -0.11019083 1.78922644 0.93495556 2.8652191  0.0000000000  0
## 61  0.36787944 -0.10748945 1.79833733 0.94311516 2.8737643  0.0000000000  0
## 62  0.38674102 -0.10491983 1.80700389 0.95087681 2.8818928  0.0000000000  0
## 63  0.40656966 -0.10247552 1.81524777 0.95825992 2.8896249  0.0000000000  0
## 64  0.42741493 -0.10015042 1.82308959 0.96528295 2.8969799  0.0000000000  0
## 65  0.44932896 -0.09793872 1.83054896 0.97196347 2.9039762  0.0000000000  0
## 66  0.47236655 -0.09583751 1.83760461 0.97814379 2.9105382  0.0000000000  0
## 67  0.49658530 -0.09383621 1.84435509 0.98419180 2.9168704  0.0000000000  0
## 68  0.52204578 -0.09193246 1.85077730 0.98994996 2.9228965  0.0000000000  0
## 69  0.54881164 -0.09012155 1.85688632 0.99542745 2.9286288  0.0000000000  0
## 70  0.57694981 -0.08839896 1.86269741 1.00063780 2.9340816  0.0000000000  0
## 71  0.60653066 -0.08676039 1.86822508 1.00559404 2.9392684  0.0000000000  0
## 72  0.63762815 -0.08520173 1.87348317 1.01030856 2.9442022  0.0000000000  0
## 73  0.67032005 -0.08371908 1.87848482 1.01479315 2.9488954  0.0000000000  0
## 74  0.70468809 -0.08230875 1.88324253 1.01905902 2.9533598  0.0000000000  0
## 75  0.74081822 -0.08096720 1.88776821 1.02311684 2.9576064  0.0000000000  0
## 76  0.77880078 -0.07969107 1.89207316 1.02697677 2.9616458  0.0000000000  0
## 77  0.81873075 -0.07847719 1.89616817 1.03064844 2.9654883  0.0000000000  0
## 78  0.86070798 -0.07732250 1.90006345 1.03414104 2.9691434  0.0000000000  0
## 79  0.90483742 -0.07622413 1.90376876 1.03746330 2.9726202  0.0000000000  0
## 80  0.95122942 -0.07517933 1.90729336 1.04062354 2.9759275  0.0000000000  0
## 81  1.00000000 -0.07418548 1.91064607 1.04362965 2.9790734  0.0000000000  0
## 82  1.05127110 -0.07317995 1.91383526 1.04648915 2.9820660  0.0000000000  0
## 83  1.10517092 -0.07198889 1.91715202 1.04923592 2.9851694  0.0000000000  0
## 84  1.16183424 -0.07088478 1.92026346 1.05187998 2.9880787  0.0000000000  0
## 85  1.22140276 -0.06983468 1.92322244 1.05439523 2.9908461  0.0000000000  0
## 86  1.28402542 -0.06883579 1.92603711 1.05678781 2.9934785  0.0000000000  0
## 87  1.34985881 -0.06788562 1.92871451 1.05906370 2.9959825  0.0000000000  0
## 88  1.41906755 -0.06698179 1.93126132 1.06122860 2.9983644  0.0000000000  0
## 89  1.49182470 -0.06612204 1.93368393 1.06328791 3.0006302  0.0000000000  0
## 90  1.56831219 -0.06530422 1.93598838 1.06524679 3.0027854  0.0000000000  0
## 91  1.64872127 -0.06457039 1.93800176 1.06683437 3.0046365  0.0000000000  0
```

```
## 92  1.73325302 -0.06378719 1.94026115 1.06887092 3.0067788  0.0000000000  0
## 93  1.82211880 -0.06312244 1.94208674 1.07031735 3.0084595  0.0000000000  0
## 94  1.91554083 -0.06245523 1.94396277 1.07189240 3.0102076  0.0000000000  0
## 95  2.01375271 -0.06181687 1.94576323 1.07342512 3.0118917  0.0000000000  0
## 96  2.11700002 -0.06138917 1.94747815 1.07488858 3.0134971 -0.0009484855  0
## 97  2.22554093 -0.06099217 1.94910809 1.07622030 3.0151624 -0.0019702280  0
## 98  2.33964685 -0.06062127 1.95065459 1.07751903 3.0166855 -0.0029380267  0
## 99  2.45960311 -0.06026835 1.95212595 1.07875705 3.0181359 -0.0038585427  0
## 100 2.58570966 -0.05993260 1.95352569 1.07993508 3.0195158 -0.0047341419  0
## 101 2.71828183 -0.05961322 1.95485719 1.08105572 3.0208285 -0.0055670334  0
##             V6         V7          V8          V9        V10        MSE
## 1   0.00000000 0.00000000 0.000000000  0.00000000 -0.5726738 33.0525220
## 2   0.01606127 0.00000000 0.000000000  0.00000000 -0.6997152 31.6363784
## 3   0.14247654 0.00000000 0.000000000  0.00000000 -0.8273290 29.8211148
## 4   0.25007137 0.00000000 0.000000000 -0.08773478 -0.9506092 27.7678900
## 5   0.34635104 0.00000000 0.000000000 -0.21319468 -1.0687806 25.7182802
## 6   0.43793547 0.00000000 0.000000000 -0.33253573 -1.1811887 23.8637191
## 7   0.52505328 0.00000000 0.000000000 -0.44605644 -1.2881147 22.1856429
## 8   0.60792231 0.00000000 0.000000000 -0.55404069 -1.3898258 20.6672568
## 9   0.68674976 0.00000000 0.000000000 -0.65675849 -1.4865764 19.2933642
## 10  0.76095500 0.00000000 0.000000000 -0.75464031 -1.5779342 17.9942234
## 11  0.82853522 0.00000000 0.000000000 -0.84842077 -1.6622148 16.6061699
## 12  0.89281391 0.00000000 0.000000000 -0.93762837 -1.7423840 15.3501773
## 13  0.95395769 0.00000000 0.000000000 -1.02248527 -1.8186433 14.2137083
## 14  1.01211945 0.00000000 0.000000000 -1.10320365 -1.8911834 13.1853885
## 15  1.06744463 0.00000000 0.000000000 -1.17998534 -1.9601857 12.2549263
## 16  1.12007157 0.00000000 0.000000000 -1.25302235 -2.0258227 11.4130094
## 17  1.17013186 0.00000000 0.000000000 -1.32249730 -2.0882586 10.6512114
## 18  1.21775068 0.00000000 0.000000000 -1.38858392 -2.1476494  9.9619080
## 19  1.26304711 0.00000000 0.000000000 -1.45144745 -2.2041437  9.3382006
## 20  1.30613440 0.00000000 0.000000000 -1.51124510 -2.2578827  8.7738468
## 21  1.34712030 0.00000000 0.000000000 -1.56812638 -2.3090009  8.2631983
## 22  1.38610729 0.00000000 0.000000000 -1.62223352 -2.3576260  7.8011444
## 23  1.41739797 0.00000000 0.000000000 -1.68172605 -2.3967205  7.3044623
## 24  1.44688531 0.03473479 0.000000000 -1.74504868 -2.4267351  6.7331004
## 25  1.47668479 0.08542614 0.000000000 -1.80624817 -2.4538270  6.1784768
## 26  1.50497436 0.13365756 0.000000000 -1.86449985 -2.4795655  5.6763118
## 27  1.53188416 0.17953673 0.000000000 -1.91991061 -2.5040487  5.2219337
## 28  1.55748155 0.22317835 0.000000000 -1.97261896 -2.5273378  4.8107954
## 29  1.58183054 0.26469154 0.000000000 -2.02275670 -2.5494910  4.4387821
## 30  1.60499202 0.30418011 0.000000000 -2.07044918 -2.5705639  4.1021705
## 31  1.62702390 0.34174280 0.000000000 -2.11581568 -2.5906090  3.7975917
## 32  1.64798127 0.37747354 0.000000000 -2.15896963 -2.6096765  3.5219975
## 33  1.66791654 0.41146166 0.000000000 -2.20001893 -2.6278141  3.2726295
## 34  1.68687955 0.44379217 0.000000000 -2.23906624 -2.6450671  3.0469920
## 35  1.70491773 0.47454590 0.000000000 -2.27620918 -2.6614787  2.8428268
## 36  1.72207617 0.50379975 0.000000000 -2.31154064 -2.6770898  2.6580904
## 37  1.73839779 0.53162687 0.000000000 -2.34514897 -2.6919396  2.4909341
## 38  1.75397912 0.55808044 0.000000000 -2.37708127 -2.7060995  2.3398705
## 39  1.76874518 0.58326017 0.000000000 -2.40749289 -2.7195347  2.2029979
## 40  1.78279073 0.60721196 0.000000000 -2.43642157 -2.7323144  2.0791494
## 41  1.79615126 0.62999561 0.000000000 -2.46393938 -2.7444709  1.9670865
## 42  1.80886019 0.65166809 0.000000000 -2.49011514 -2.7560345  1.8656879
## 43  1.82094930 0.67228359 0.000000000 -2.51501428 -2.7670341  1.7739386
```

```
## 44  1.83244882 0.69189366 0.000000000 -2.53869908 -2.7774973  1.6909204
## 45  1.84338750 0.71054734 0.000000000 -2.56122876 -2.7874501  1.6158024
## 46  1.85379270 0.72829126 0.000000000 -2.58265966 -2.7969176  1.5478329
## 47  1.86369042 0.74516981 0.000000000 -2.60304536 -2.8059233  1.4863315
## 48  1.87310543 0.76122517 0.000000000 -2.62243683 -2.8144898  1.4306827
## 49  1.88206126 0.77649751 0.000000000 -2.64088257 -2.8226385  1.3803297
## 50  1.89058032 0.79102501 0.000000000 -2.65842870 -2.8303898  1.3347683
## 51  1.89868389 0.80484399 0.000000000 -2.67511910 -2.8377631  1.2935427
## 52  1.90639225 0.81798902 0.000000000 -2.69099550 -2.8447767  1.2562402
## 53  1.91372466 0.83049295 0.000000000 -2.70609759 -2.8514484  1.2224875
## 54  1.92069947 0.84238706 0.000000000 -2.72046315 -2.8577946  1.1919468
## 55  1.92733412 0.85370108 0.000000000 -2.73412809 -2.8638313  1.1643125
## 56  1.93364519 0.86446332 0.000000000 -2.74712659 -2.8695736  1.1393079
## 57  1.93964846 0.87470067 0.000000000 -2.75949114 -2.8750358  1.1166828
## 58  1.94535895 0.88443874 0.000000000 -2.77125266 -2.8802317  1.0962108
## 59  1.95079094 0.89370189 0.000000000 -2.78244057 -2.8851741  1.0776869
## 60  1.95595801 0.90251326 0.000000000 -2.79308284 -2.8898755  1.0609258
## 61  1.96087307 0.91089489 0.000000000 -2.80320607 -2.8943476  1.0457598
## 62  1.96554843 0.91886775 0.000000000 -2.81283559 -2.8986017  1.0320369
## 63  1.96999576 0.92645177 0.000000000 -2.82199548 -2.9026482  1.0196200
## 64  1.97422620 0.93366591 0.000000000 -2.83070863 -2.9064974  1.0083847
## 65  1.97825031 0.94052822 0.000000000 -2.83899684 -2.9101588  0.9982186
## 66  1.98214124 0.94703806 0.000000000 -2.84683488 -2.9136831  0.9890780
## 67  1.98578141 0.95324768 0.000000000 -2.85433518 -2.9169954  0.9807508
## 68  1.98924204 0.95915495 0.000000000 -2.86147106 -2.9201449  0.9732145
## 69  1.99253382 0.96477414 0.000000000 -2.86825896 -2.9231407  0.9663952
## 70  1.99566506 0.97011928 0.000000000 -2.87471582 -2.9259905  0.9602250
## 71  1.99864359 0.97520373 0.000000000 -2.88085777 -2.9287012  0.9546419
## 72  2.00147686 0.98004021 0.000000000 -2.88670017 -2.9312798  0.9495901
## 73  2.00417194 0.98464081 0.000000000 -2.89225764 -2.9337326  0.9450190
## 74  2.00673559 0.98901704 0.000000000 -2.89754406 -2.9360658  0.9408829
## 75  2.00917420 0.99317984 0.000000000 -2.90257266 -2.9382852  0.9371405
## 76  2.01149388 0.99713961 0.000000000 -2.90735602 -2.9403963  0.9337542
## 77  2.01370043 1.00090627 0.000000000 -2.91190609 -2.9424045  0.9306901
## 78  2.01579937 1.00448922 0.000000000 -2.91623425 -2.9443147  0.9279176
## 79  2.01779593 1.00789743 0.000000000 -2.92035132 -2.9461318  0.9254089
## 80  2.01969513 1.01113942 0.000000000 -2.92426760 -2.9478603  0.9231390
## 81  2.02150170 1.01422330 0.000000000 -2.92799288 -2.9495044  0.9210851
## 82  2.02322016 1.01715677 0.001193205 -2.93168396 -2.9509554  0.9191095
## 83  2.02472779 1.02070014 0.004530535 -2.93558106 -2.9520182  0.9169930
## 84  2.02617810 1.02389556 0.007664960 -2.93926445 -2.9530494  0.9151025
## 85  2.02755755 1.02693471 0.010646276 -2.94276821 -2.9540304  0.9133921
## 86  2.02886971 1.02982563 0.013482190 -2.94610108 -2.9549635  0.9118444
## 87  2.03011789 1.03257557 0.016179795 -2.94927141 -2.9558511  0.9104440
## 88  2.03130519 1.03519139 0.018745836 -2.95228711 -2.9566954  0.9091769
## 89  2.03243458 1.03767964 0.021186730 -2.95515574 -2.9574986  0.9080304
## 90  2.03350890 1.04004653 0.023508580 -2.95788447 -2.9582625  0.9069929
## 91  2.03467907 1.04214875 0.025534483 -2.96037148 -2.9590937  0.9061070
## 92  2.03550773 1.04443632 0.027813473 -2.96294530 -2.9596841  0.9052064
## 93  2.03656256 1.04634121 0.029650346 -2.96519891 -2.9604332  0.9044797
## 94  2.03745544 1.04826916 0.031537596 -2.96742357 -2.9610679  0.9037858
## 95  2.03828805 1.05011715 0.033351295 -2.96955174 -2.9616599  0.9031531
## 96  2.03897959 1.05184275 0.035084894 -2.97152386 -2.9622317  0.9025428
## 97  2.03961952 1.05347539 0.036736271 -2.97340285 -2.9627807  0.9019820
```

```
## 98  2.04023405 1.05502500 0.038299608 -2.97518579 -2.9633018  0.9014769
## 99  2.04081713 1.05649963 0.039787885 -2.97688269 -2.9637966  0.9010196
## 100 2.04137159 1.05790248 0.041203771 -2.97849696 -2.9642673  0.9006057
## 101 2.04189898 1.05923693 0.042550635 -2.98003253 -2.9647149  0.9002313
```

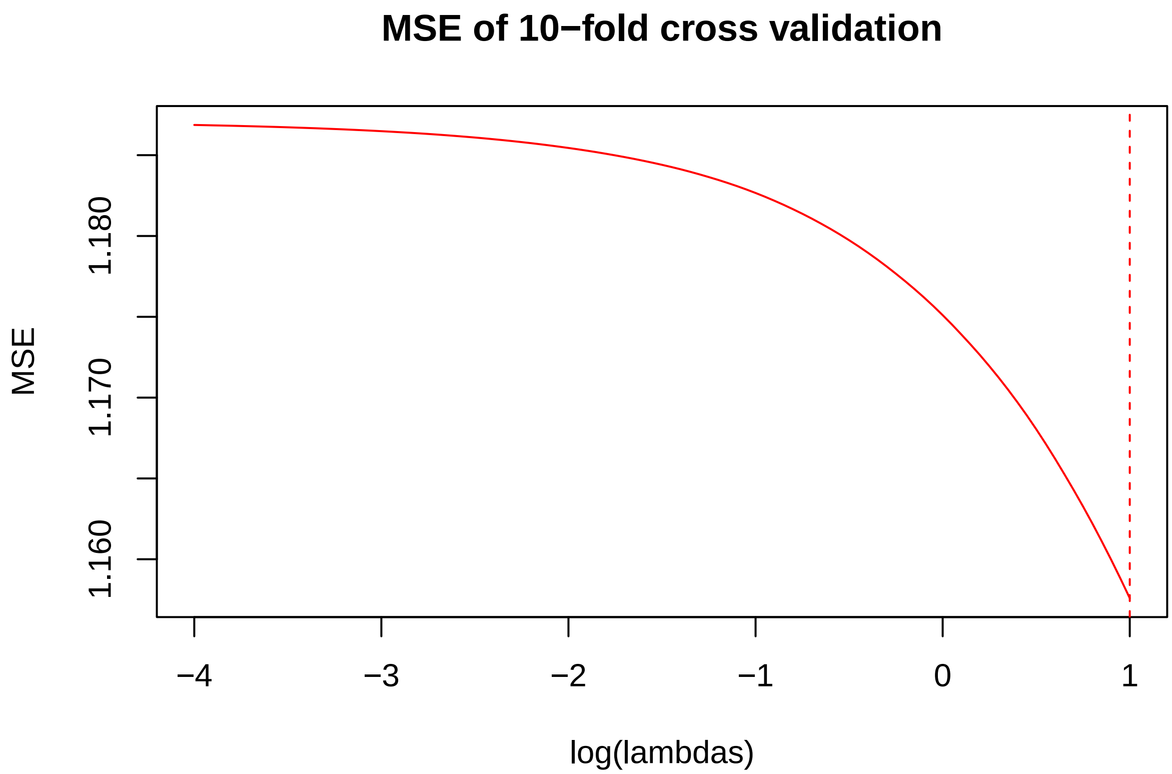
The higher the regularization parameter lambda is, the lower the MSE is for the respective coefficients of the glmnet function. This is not so strongly the case for our own algorithm, suspiciously, the MSE is always almost the same.

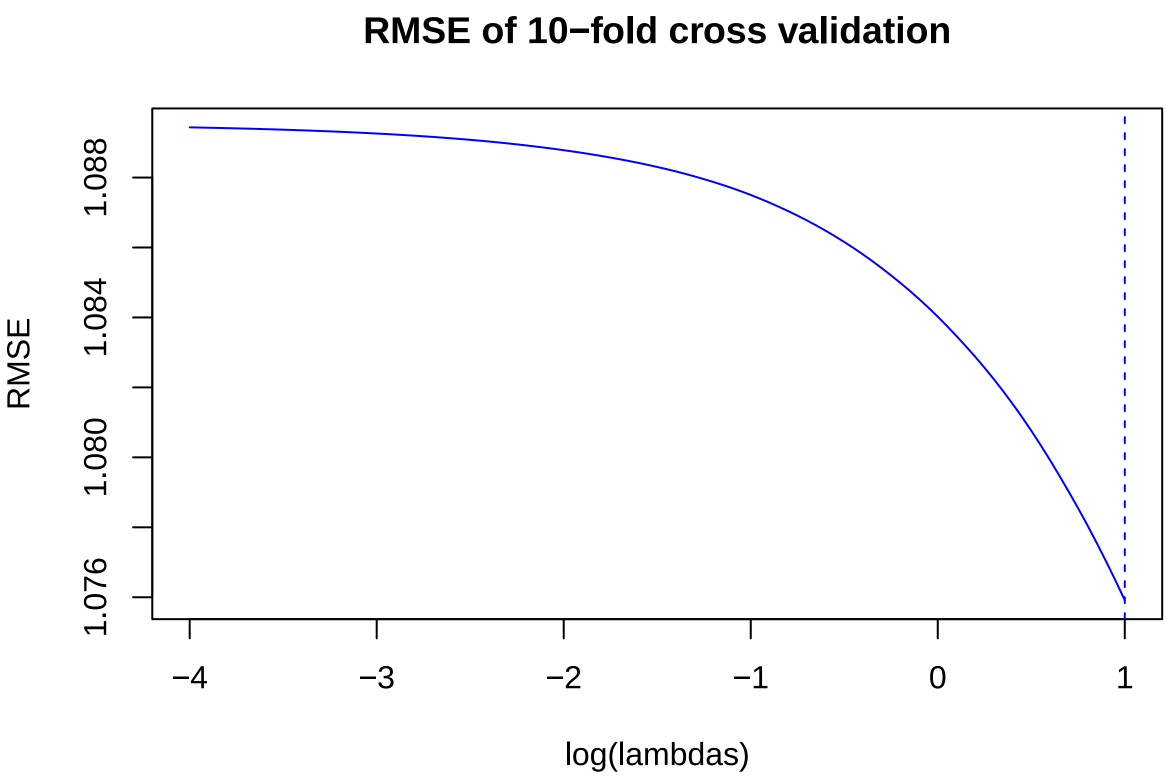
Writing a custom cross-validation function

```r
cv.lasso <- function(X, y, lambdas, k=10, tolerance_limit=1e-07, max_iter=1e+05, verbose=F){
  n <- nrow(X)
  # this vector is the random fold each datapoint is assigned to
  fold_assignments <- sample(rep(1:k, length.out=n))

  # create anempty matrix for all MSE results
  mse_results <- matrix(0, nrow=length(lambdas), ncol=k)

  # do the k-fold CV
  # iterate over each of the k folds: fold
  for (fold in 1:k){
    # split data into training and testing sets
    # if an observation is part of the current fold, it is val, otherwise train
    train_idx <- which(fold_assignments != fold)
    val_idx <- which(fold_assignments == fold)
    X_train <- X[train_idx,]
    y_train <- y[train_idx,]
    X_val <- X[val_idx,]
    y_val <- y[val_idx,]

    # apply our custom lasso shooting algorithm
    for (l in seq_along(lambdas)){
      # fit the model (get the coefficients) using train
      beta <- lasso_shooting(X_train, y_train, lambdas[l], tolerance_limit, max_iter, verbose)
      # evaluate using the val data
      mse <- evaluate(y_val, X_val, beta)
      # save the results for this fold
      mse_results[l, fold] <- mse
    }
  }
  # now find the lambda that minimizes the average mse & rmse across all folds
  mse.means <- rowMeans(mse_results)
  rmse.means <- sqrt(mse.means)
  lambda.min.mse <- lambdas[which.min(mse.means)]
  lambda.min.rmse <- lambdas[which.min(rmse.means)]

  # plot the results
  plot(log(lambdas), mse.means, ylab = "MSE", main="MSE of 10-fold cross validation",
       type="l", col="red")
  abline(v=log(lambda.min.mse), col="red", lty=2)
  plot(log(lambdas), rmse.means, ylab = "RMSE", main="RMSE of 10-fold cross validation",
       type="l", col="blue")
  abline(v=log(lambda.min.rmse), col="blue", lty=2)
```

```
  # Return results
  list(
    lambdas = lambdas,
    avg_mse = mse.means,
    avg_rmse = rmse.means,
    lambda_min_mse = lambda.min.mse,
    lambda_min_rmse = lambda.min.rmse
  )
}
```

```
cv.lasso(X, y, lambdas)
```






```
## $lambdas
##   [1] 0.01831564 0.01925470 0.02024191 0.02127974 0.02237077 0.02351775
##   [7] 0.02472353 0.02599113 0.02732372 0.02872464 0.03019738 0.03174564
##  [13] 0.03337327 0.03508435 0.03688317 0.03877421 0.04076220 0.04285213
##  [19] 0.04504920 0.04735892 0.04978707 0.05233971 0.05502322 0.05784432
##  [25] 0.06081006 0.06392786 0.06720551 0.07065121 0.07427358 0.07808167
##  [31] 0.08208500 0.08629359 0.09071795 0.09536916 0.10025884 0.10539922
##  [37] 0.11080316 0.11648416 0.12245643 0.12873490 0.13533528 0.14227407
##  [43] 0.14956862 0.15723717 0.16529889 0.17377394 0.18268352 0.19204991
##  [49] 0.20189652 0.21224797 0.22313016 0.23457029 0.24659696 0.25924026
##  [55] 0.27253179 0.28650480 0.30119421 0.31663677 0.33287108 0.34993775
##  [61] 0.36787944 0.38674102 0.40656966 0.42741493 0.44932896 0.47236655
##  [67] 0.49658530 0.52204578 0.54881164 0.57694981 0.60653066 0.63762815
##  [73] 0.67032005 0.70468809 0.74081822 0.77880078 0.81873075 0.86070798
##  [79] 0.90483742 0.95122942 1.00000000 1.05127110 1.10517092 1.16183424
##  [85] 1.22140276 1.28402542 1.34985881 1.41906755 1.49182470 1.56831219
##  [91] 1.64872127 1.73325302 1.82211880 1.91554083 2.01375271 2.11700002
##  [97] 2.22554093 2.33964685 2.45960311 2.58570966 2.71828183
##
## $avg_mse
##   [1] 1.186869 1.186857 1.186845 1.186833 1.186819 1.186805 1.186790 1.186775
##   [9] 1.186759 1.186742 1.186724 1.186705 1.186685 1.186664 1.186642 1.186619
##  [17] 1.186595 1.186569 1.186542 1.186514 1.186485 1.186453 1.186421 1.186386
##  [25] 1.186350 1.186312 1.186272 1.186231 1.186187 1.186140 1.186092 1.186040
##  [33] 1.185987 1.185930 1.185871 1.185809 1.185743 1.185674 1.185602 1.185526
##  [41] 1.185446 1.185363 1.185276 1.185185 1.185089 1.184988 1.184882 1.184771
##  [49] 1.184654 1.184532 1.184403 1.184267 1.184125 1.183974 1.183814 1.183646
##  [57] 1.183469 1.183284 1.183088 1.182880 1.182662 1.182429 1.182185 1.181927
##  [65] 1.181658 1.181375 1.181078 1.180767 1.180440 1.180097 1.179738 1.179362
```

```
##  [73] 1.178967 1.178553 1.178120 1.177666 1.177199 1.176710 1.176199 1.175664
##  [81] 1.175104 1.174517 1.173904 1.173275 1.172623 1.171941 1.171228 1.170484
##  [89] 1.169708 1.168898 1.168046 1.167158 1.166234 1.165271 1.164288 1.163271
##  [97] 1.162215 1.161117 1.159983 1.158808 1.157593
##
## $avg_rmse
##   [1] 1.089435 1.089430 1.089424 1.089418 1.089412 1.089406 1.089399 1.089392
##   [9] 1.089385 1.089377 1.089368 1.089360 1.089351 1.089341 1.089331 1.089320
##  [17] 1.089309 1.089298 1.089285 1.089272 1.089259 1.089244 1.089229 1.089214
##  [25] 1.089197 1.089180 1.089161 1.089142 1.089122 1.089101 1.089078 1.089055
##  [33] 1.089030 1.089004 1.088977 1.088948 1.088918 1.088887 1.088853 1.088819
##  [41] 1.088782 1.088744 1.088704 1.088662 1.088618 1.088572 1.088523 1.088472
##  [49] 1.088418 1.088362 1.088303 1.088241 1.088175 1.088106 1.088032 1.087955
##  [57] 1.087874 1.087789 1.087699 1.087603 1.087502 1.087396 1.087283 1.087165
##  [65] 1.087041 1.086911 1.086774 1.086631 1.086481 1.086323 1.086158 1.085984
##  [73] 1.085803 1.085612 1.085412 1.085203 1.084988 1.084763 1.084527 1.084280
##  [81] 1.084022 1.083752 1.083469 1.083178 1.082877 1.082562 1.082233 1.081889
##  [89] 1.081530 1.081156 1.080762 1.080351 1.079923 1.079477 1.079022 1.078550
##  [97] 1.078061 1.077551 1.077025 1.076479 1.075915
##
## $lambda_min_mse
## [1] 2.718282
##
## $lambda_min_rmse
## [1] 2.718282
```

Task 2

Splitting data

```r
N <- nrow(Hitters)
Hitters <- Hitters %>%
  mutate_all(as.numeric) %>%
  replace(is.na(.), 0)

train.idx <- sample(1:N, round(N*0.7))
train <- Hitters[train.idx, ] %>% as.matrix()
test <- Hitters[-train.idx, ] %>% as.matrix()
rownames(train) <- NULL
rownames(test) <- NULL
#
train_X <- train[, -19]
train_y <- train[, 19]
test_X <- test[, -19]
test_y <- test[, 19]
```

Fitting the shooting algorithm

```r
# lasso_coeffs <- get_lasso_coeffs(train_X, train_y, lambdas)

# lasso_shooting(train_X, y, 1)
```

Fitting glmnet lasso

```
lasso.fit <- cv.glmnet(train_X, train_y, alpha=1)
lambda_min <- lasso.fit$lambda.min
beta.lasso <- coef(lasso.fit, s = lambda_min ) %>% as.numeric()%>% print()
```

```
##  [1]  16.85550841   0.00000000   2.42676048   0.00000000   0.00000000
##  [6]   0.00000000   1.45285131   0.00000000   0.00000000   0.08129135
## [11]   0.00000000   0.00000000   0.32369867   0.00000000  21.29363597
## [16] -88.84613484   0.15323306   0.12670661   0.00000000   0.00000000
```

Fitting glmnet ridge

```
ridge.fit <- cv.glmnet(train_X, train_y, alpha=0)
lambda_min <- ridge.fit$lambda.min
beta.ridge <- coef(ridge.fit, s = lambda_min ) %>% as.numeric() %>% print()
```

```
##  [1]  1.137722e+02 -6.263377e-01  3.479886e+00 -1.542190e+00 -1.255491e+00
##  [6]  1.577530e+00  3.491678e+00 -1.540967e+01  4.849526e-03  1.680950e-01
## [11]  3.818038e-01  2.523794e-01  2.543787e-01 -3.640202e-01  8.776691e+01
## [16] -1.066474e+02  1.844457e-01  3.845088e-01 -2.953881e+00 -5.657960e+01
```

Fitting ordinary least squares

```
ols.fit <- cv.glmnet(train_X, train_y, alpha=0, lambdas=c(0))
lambda_min <- ols.fit$lambda.min
beta.ols <- coef(ols.fit, s = lambda_min ) %>% as.numeric()%>% print()
```

```
##  [1]  9.714388e+01 -4.905729e-01  3.021456e+00 -1.761473e+00 -9.139812e-01
##  [6]  1.623012e+00  3.108230e+00 -1.377127e+01  6.167217e-03  1.511235e-01
## [11]  3.787985e-01  2.283846e-01  2.391024e-01 -3.056357e-01  8.162023e+01
## [16] -1.049300e+02  1.846052e-01  3.636829e-01 -2.746463e+00 -4.905525e+01
```

Comparing their fit results

```
data.frame(
  model=c("Lasso", "Ridge", "Ordinary Least Squares"),
  MSE=c(
  evaluate(test_y, test_X, beta.lasso),
  evaluate(test_y, test_X, beta.ridge),
  evaluate(test_y, test_X, beta.ols)
  )
) %>%
  mutate(RMSE = sqrt(MSE))
```

```
##                    model      MSE     RMSE
## 1                  Lasso 109057.0 330.2377
## 2                  Ridge 107240.3 327.4756
## 3 Ordinary Least Squares 106845.3 326.8720
```

Ordinary least squares performed worst out of the three models. Ridge was sligthly better than Lasso though.

As expected, the lasso algorithm set some (5) coefficients all the way down to zero, whereas Ridge had no coefficients equal to zero in the end, just as in the ordinary least squares.